Numerical Algorithm for the Topology of Singular Plane Curves

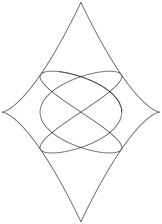
George Krait   Sylvain Lazard   Guillaume Moroz   Marc Pouget

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

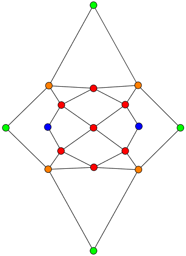
JGA 2019

Problem

To draw plane curves with the correct topology, preserving singular points' location and distinguishing their different types.



Given curve

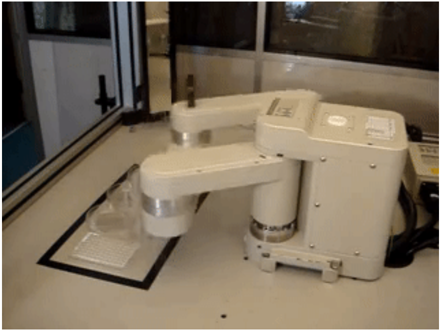


Topologically-correct graph

Example

Robot

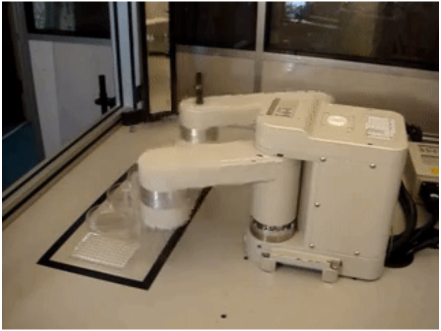
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

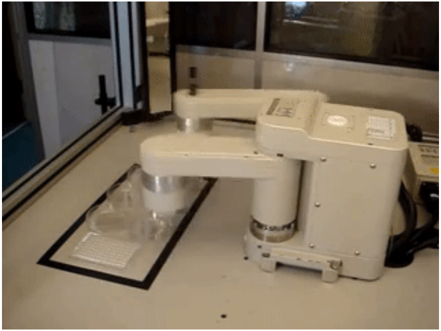
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

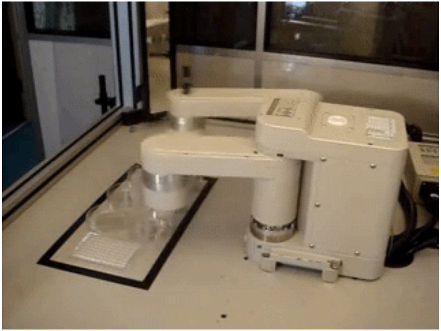
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

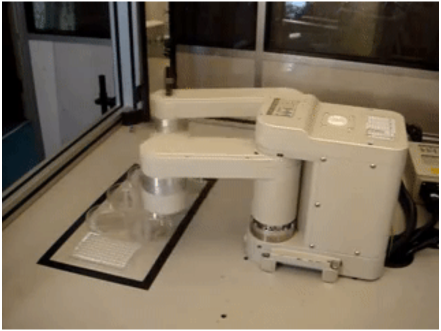
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

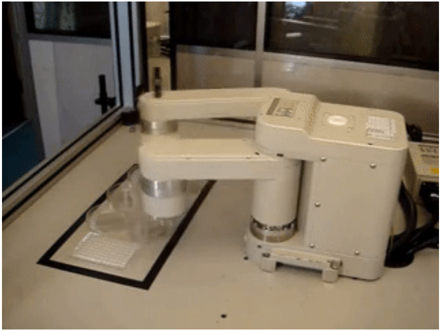
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

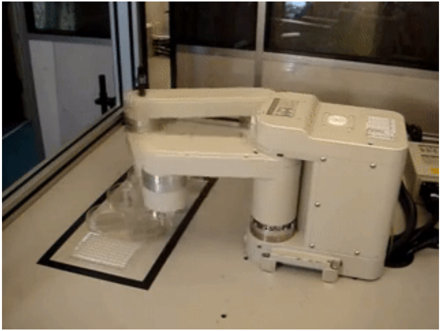
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

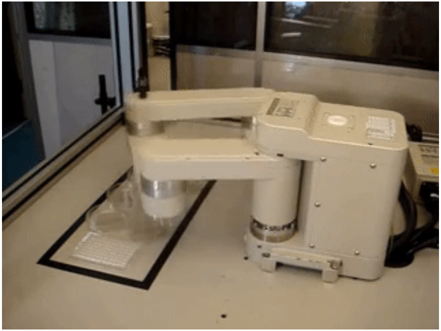
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

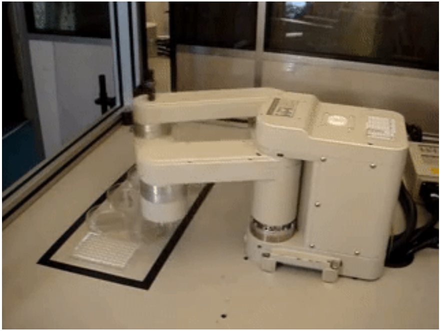
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

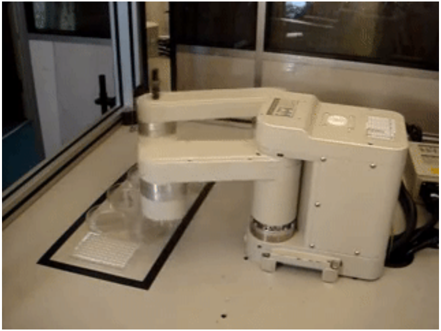
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

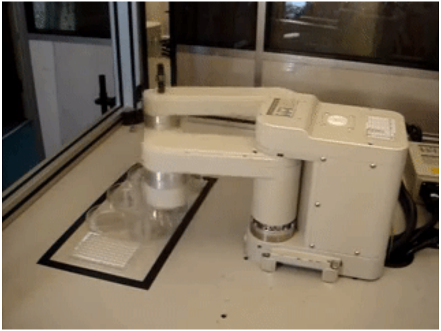
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

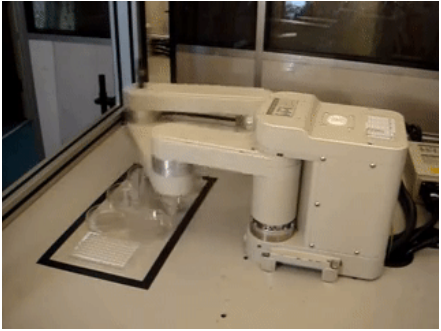
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

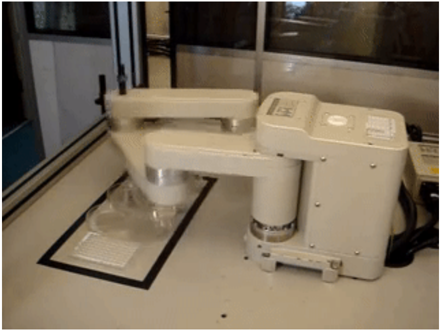
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

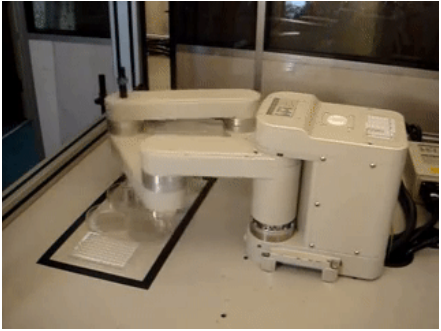
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

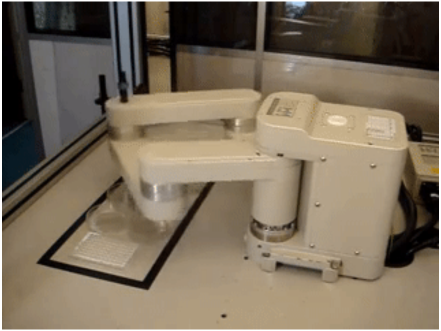
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables

Example

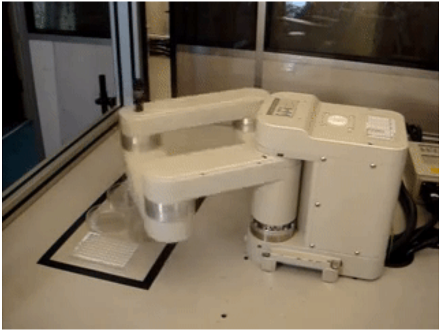
Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables

Example

Robot

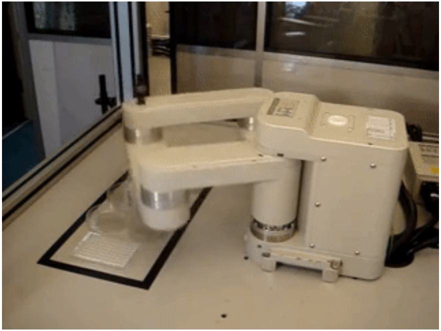
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

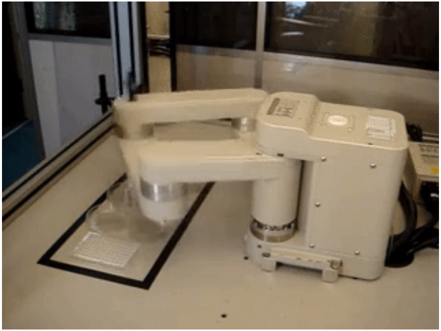
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

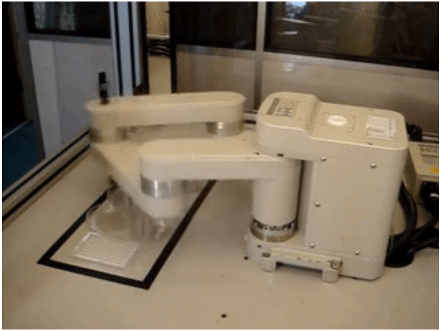
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

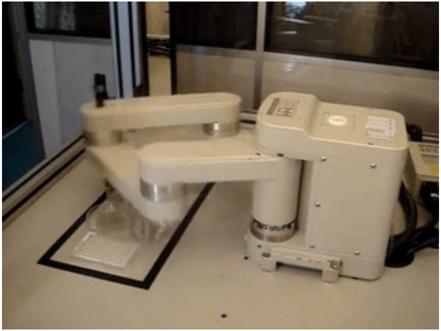
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

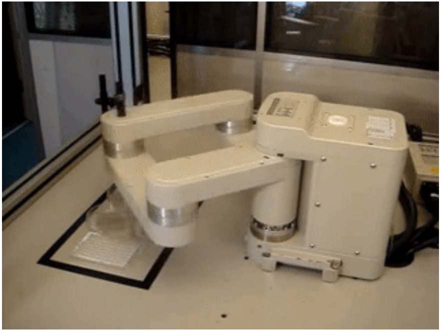
Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables

Example

Robot

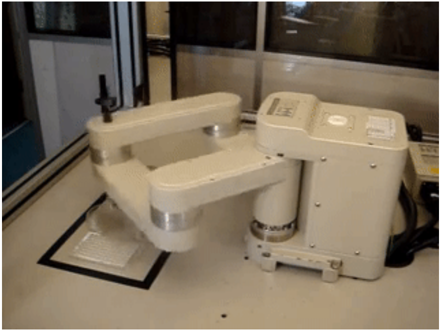
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

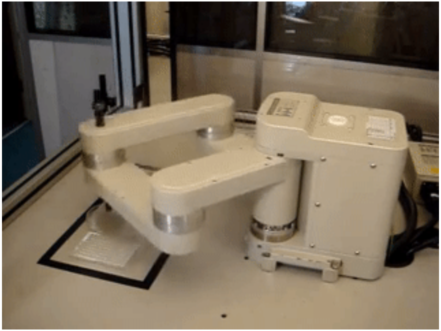
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

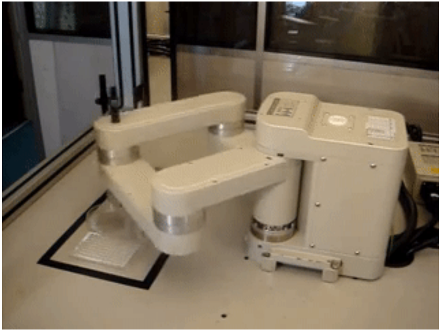
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

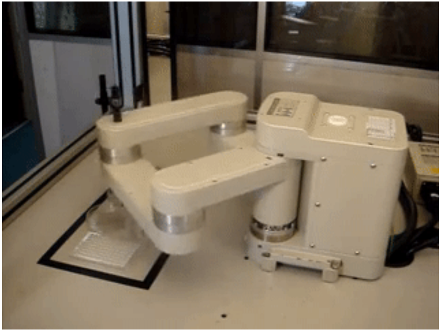
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

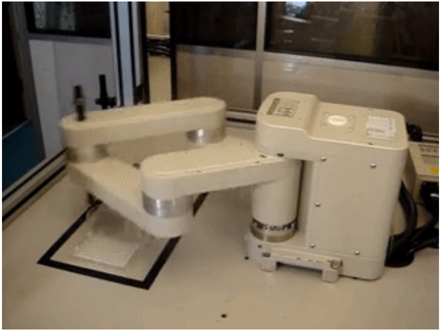
Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables

Example

Robot

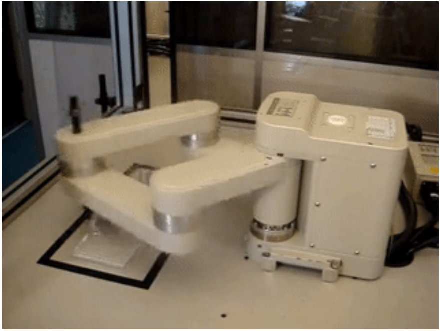
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

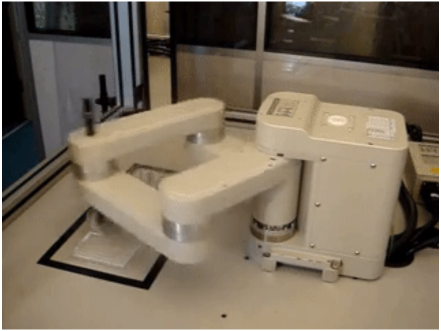
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

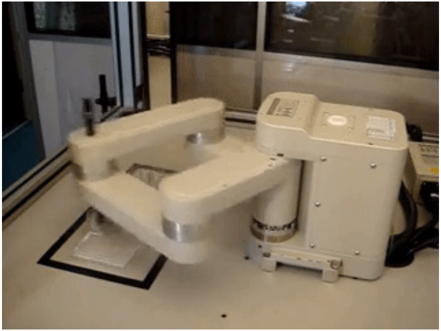
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

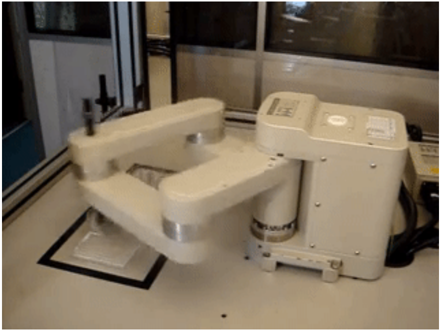
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables

Example

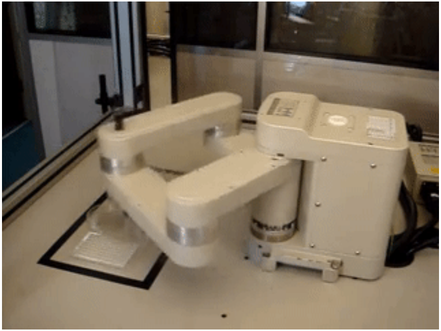
Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables

Example

Robot

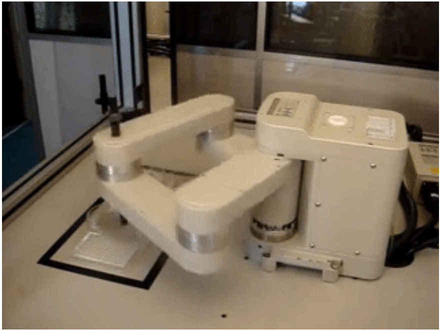
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables

Example

Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables

Example

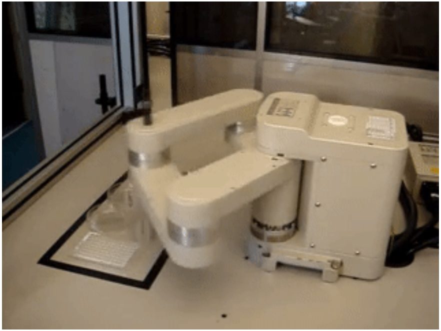
Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables

Example

Robot

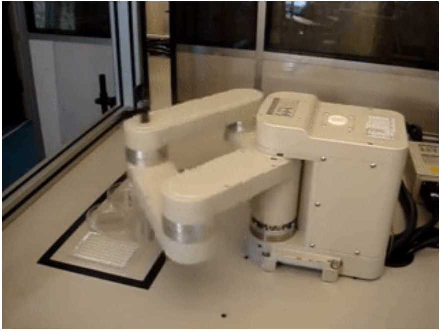
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

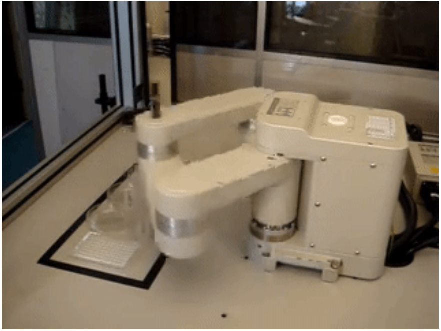
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

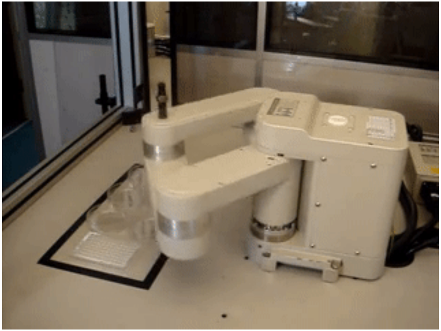
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

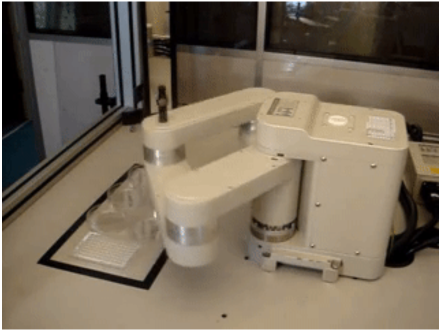
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

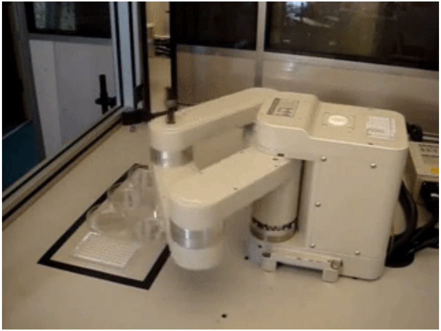
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

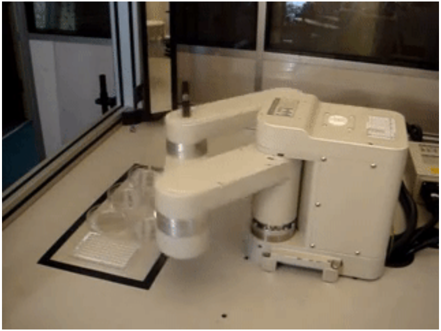
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

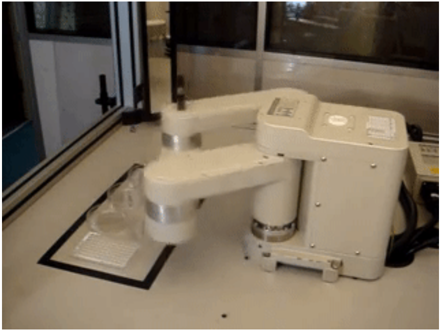
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables

Example

Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables

Example

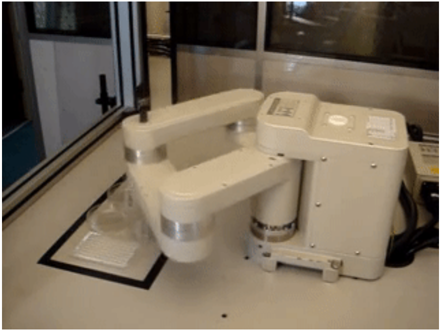
Robot

- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables

Example

Robot

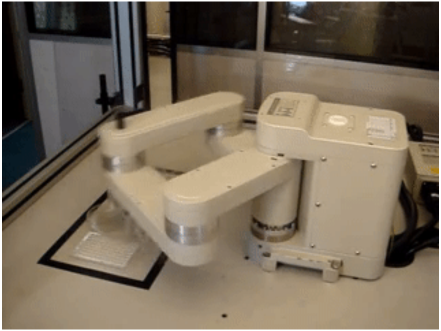
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

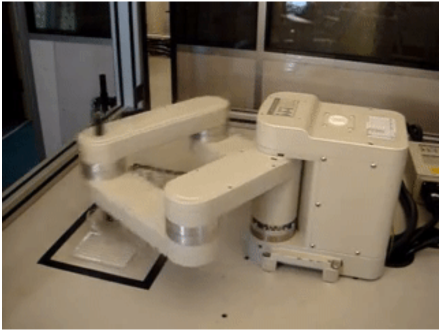
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

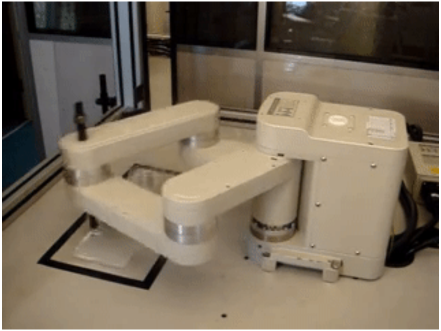
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

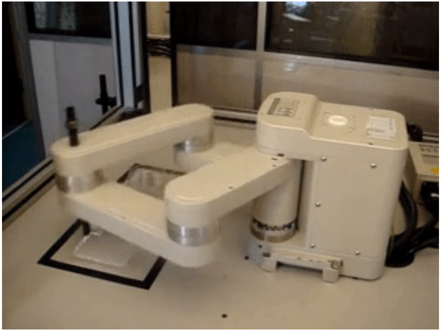
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

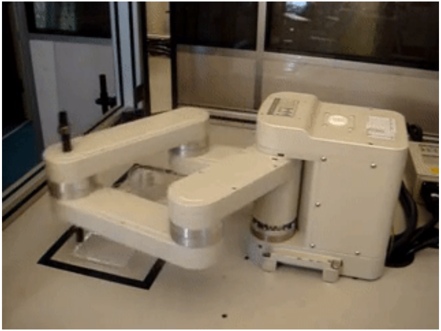
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

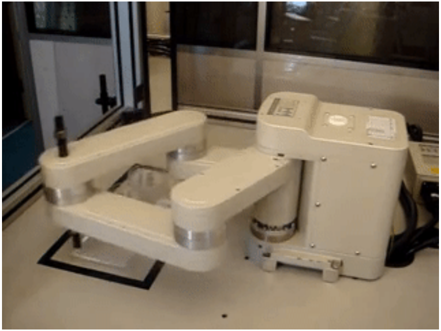
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

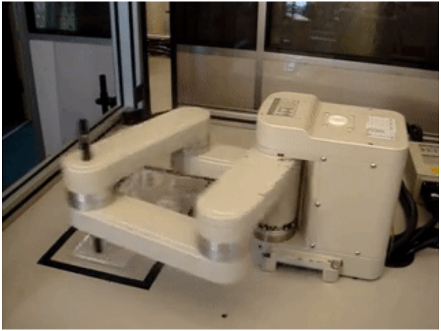
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

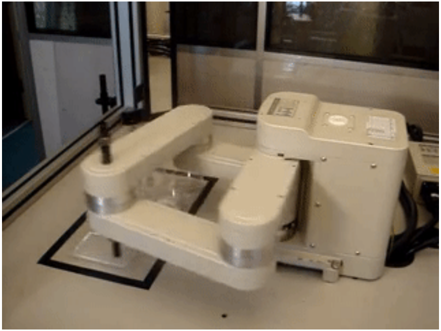
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

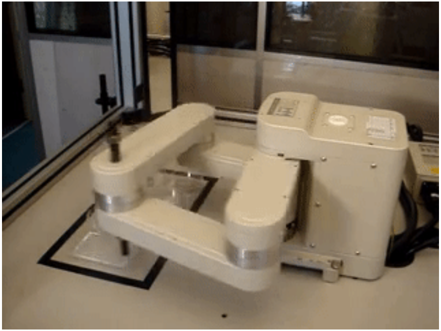
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

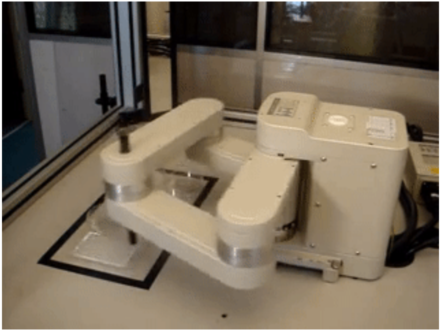
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

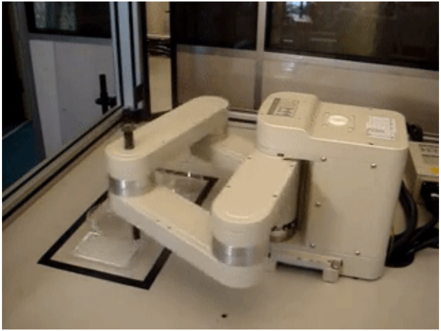
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

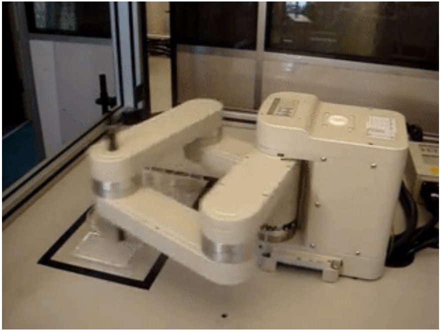
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

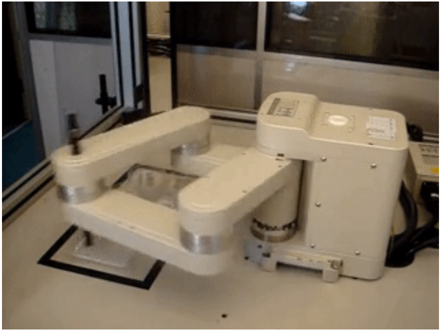
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

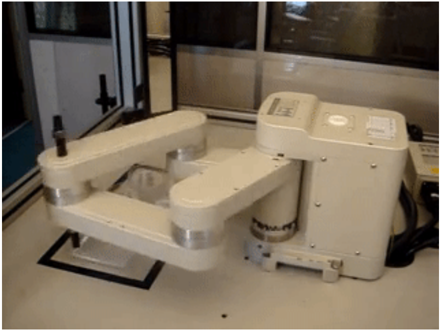
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

Robot

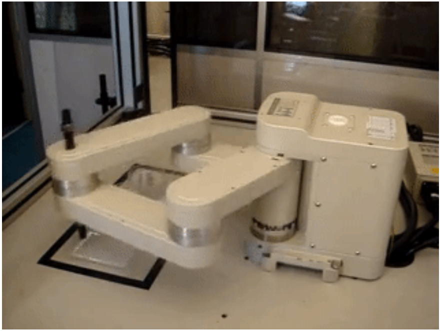
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Example

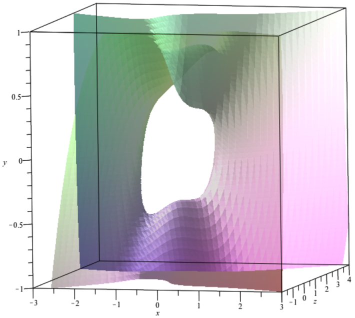
Robot

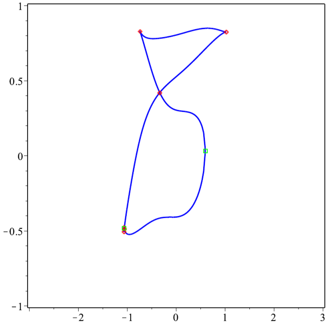
- k_1 links
- k_2 joints
- $n = k_1 + k_2$
- 2 motors in 2 joints $\rightarrow$ 2 control variables



Geometric modeling of the motion: a surface in n-dim space

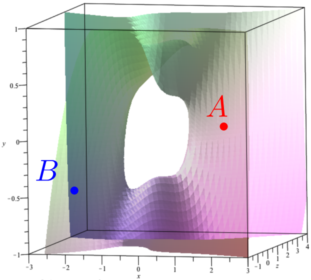
Geometric modeling of the motion: a surface in n-dim space



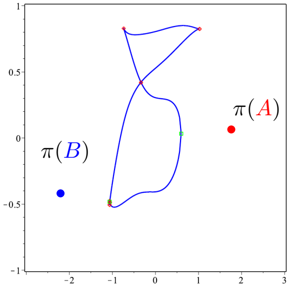


Left: a surface $f(x, y, z) = 0$. Its (smooth) silhouette curve $f = \frac{\partial f}{\partial z} = 0$.
Right: the projection of the silhouette is singular with node and cusp singularities.

Geometric modeling of the motion: surface in *n*-dim space

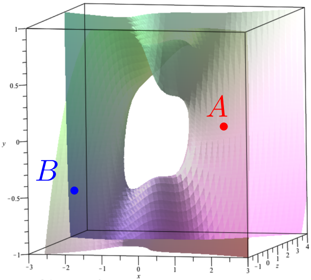


Configuration space

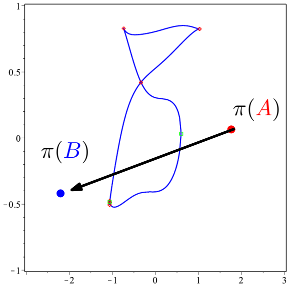


Control variables

Geometric modeling of the motion: surface in *n*-dim space

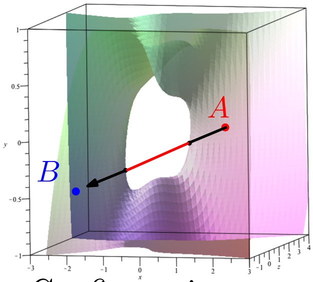


Configuration space

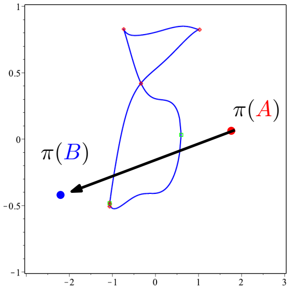


Control variables

Geometric modeling of the motion: surface in n-dim space

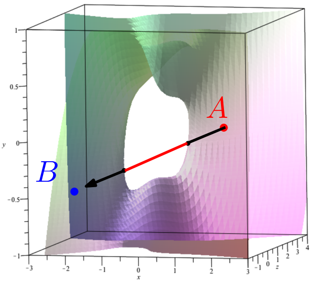


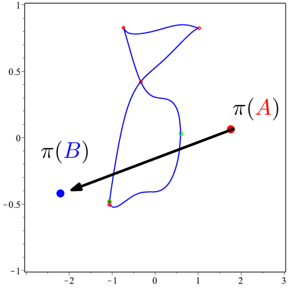
Configuration space

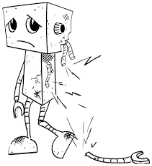


Control variables

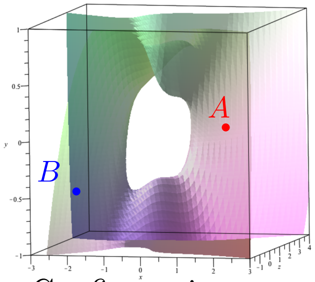
Geometric modeling of the motion: surface in n-dim space



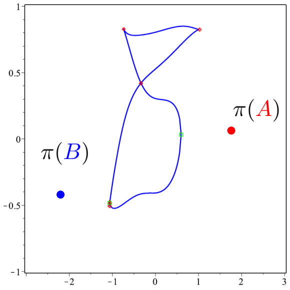




Geometric modeling of the motion: surface in n-dim space

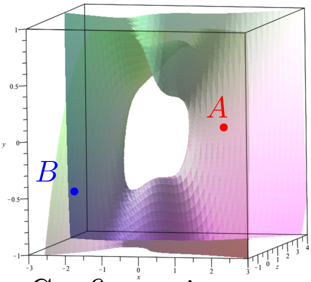


Configuration space

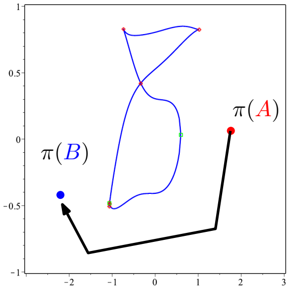


Control variables

Geometric modeling of the motion: surface in *n*-dim space

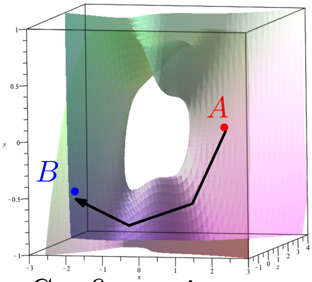


Configuration space

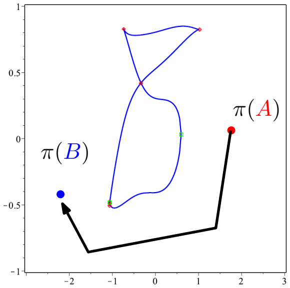


Control variables

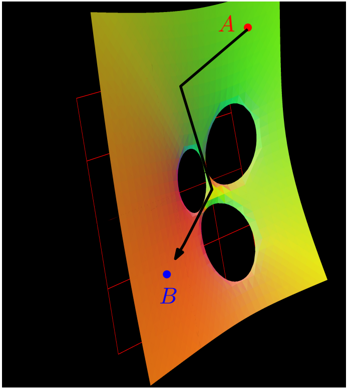
Geometric modeling of the motion: surface in *n*-dim space

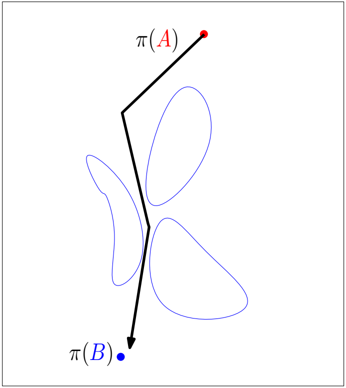


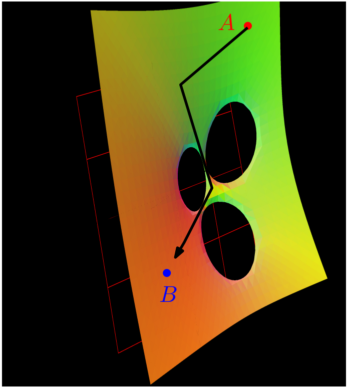
Configuration space

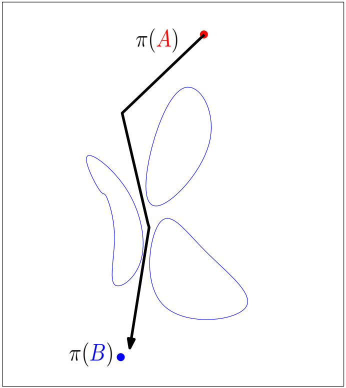


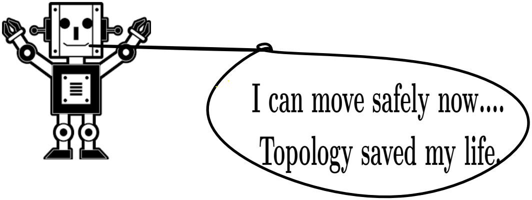
Control variables


A
B


$\pi(A)$
$\pi(B)$


A
B


$\pi(A)$
$\pi(B)$


I can move safely now....
Topology saved my life.

Previous work

Question

How to compute the topology of a curve?

Previous work

Question

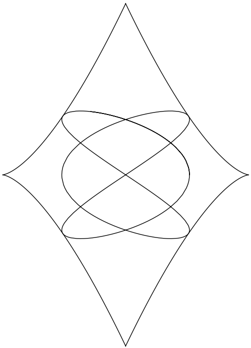
How to compute the topology of a curve?

1. Smooth curve:
   global subdivision (Snyder, 1992)
   (Plantinga and Vegter, 2004)
   (Liang et al., 2008)
   (Lin and Yap, 2011)
   continuation approaches (Beltrán and Leykin, 2013)

Question

How to compute the topology of a curve?

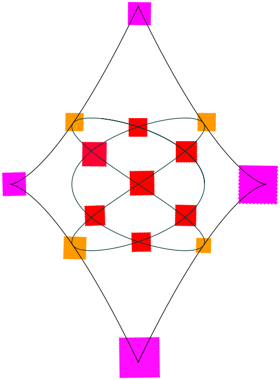
2. Singular curve:



Question

How to compute the topology of a curve?

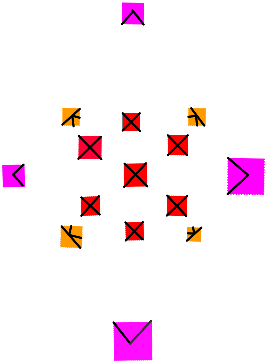
2. Singular curve:
   2.1. Isolate singular points, distinguishing their different types.



Question

How to compute the topology of a curve?

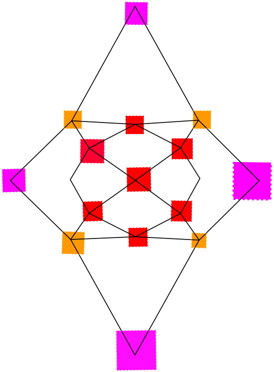
2. Singular curve:
   2.2. Compute the topology in neighborhoods of the singular points.



Question

How to compute the topology of a curve?

2. Singular curve:
    2.3. Compute the topology in the remaining smooth part.



Main challenge

It is difficult to isolate singular points.

Main challenge

It is difficult to isolate singular points.

Available software to isolate singularities:

Main challenge

It is difficult to isolate singular points.

Available software to isolate singularities:

- Symbolic methods: Certified... not efficient

Main challenge

It is difficult to isolate singular points.

Available software to isolate singularities:

- Symbolic methods: Certified... not efficient
- Numerical methods: Efficient... not certified

Main challenge

It is difficult to isolate singular points.

Available software to isolate singularities:

- Symbolic methods: Certified... not efficient
- Numerical methods: Efficient... not certified

Our main goal is to combine the benefits of both.

How?... Certified numerical methods (Interval Newton Method)

⚠ *Condition* $\longrightarrow$ certified numerical methods

Interval Newton method: square regular system

Square: the number of variables = the number of equations.
Regular: the Jacobian matrix is full rank.

⚠ Condition → certified numerical methods

Interval Newton method: square regular system

For our problem:

We describe the singularities using a square regular system.

Square: the number of variables = the number of equations.
Regular: the Jacobian matrix is full rank.

The usual system that describes singularities

Given curve $f(x, y) = 0$

The usual system that describes singularities

Given curve $f(x, y) = 0$

Its singularities:

$$\left\{ \begin{array}{l} f(x,y) = 0 \\ \frac{\partial f}{\partial x}(x,y) = 0 \\ \frac{\partial f}{\partial y}(x,y) = 0 \end{array} \right\}$$

The usual system that describes singularities

Given curve $f(x, y) = 0$

Its singularities:

$$\left\{ \begin{array}{l} f(x, y) = 0 \\ \frac{\partial f}{\partial x}(x, y) = 0 \\ \frac{\partial f}{\partial y}(x, y) = 0 \end{array} \right\}$$

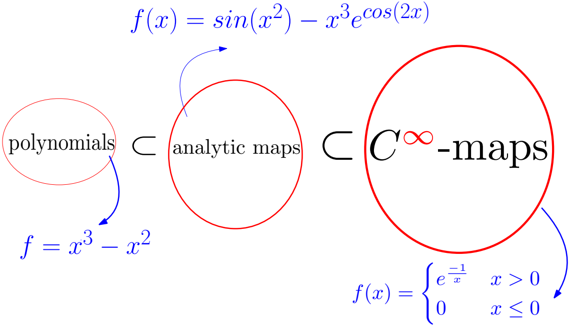
Not square!

Our approach

We restrict ourselves to the plane projection of smooth curves in higher dimensions.

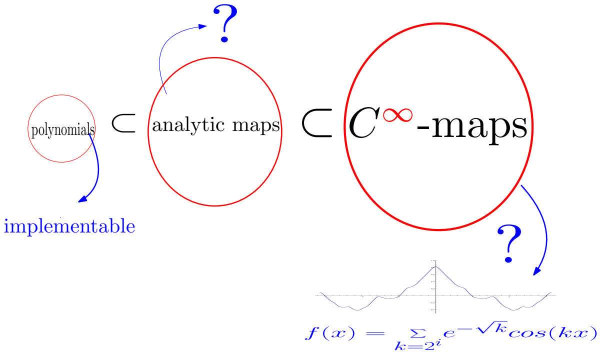
Recall

C^∞-map: differentiable ∞ times



Recall

C^∞-map: differentiable ∞ times



Assumptions

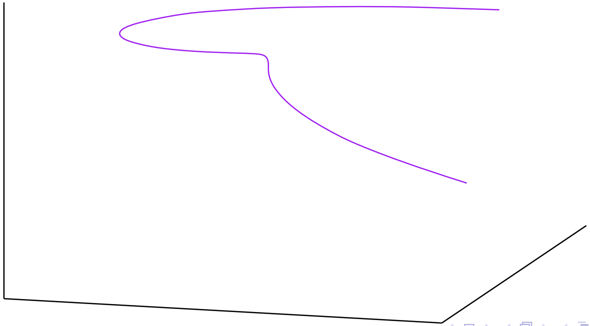
$\mathfrak{C}_n \subset \mathbb{R}^n$: zero set of $n-1$ C^∞-maps

Assumptions

$\mathfrak{C}_n \subset \mathbb{R}^n$: zero set of $n-1$ C^∞-maps | $\mathfrak{C}$ the plane projection of $\mathfrak{C}_n$

such that:

- $\mathfrak{C}_n$ smooth (actually, full-rank Jacobian).

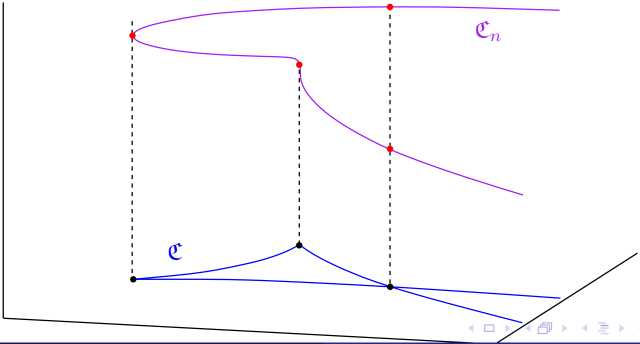


Assumptions

$\mathfrak{C}_n \subset \mathbb{R}^n$: zero set of $n-1$ C^∞-maps | $\mathfrak{C}$ the plane projection of $\mathfrak{C}_n$

such that:

- $\mathfrak{C}_n$ smooth (actually, full-rank Jacobian).
- Bad points set is finite.

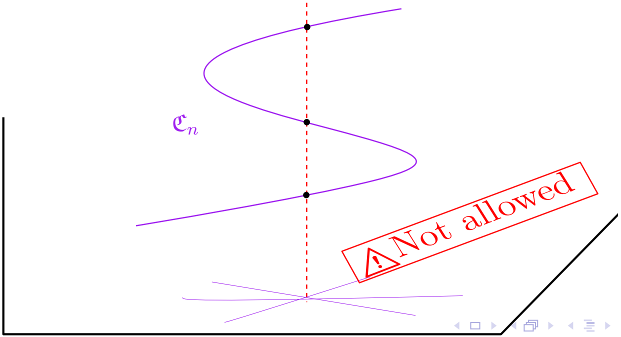


Assumptions

$\mathfrak{C}_n \subset \mathbb{R}^n$: zero set of $n-1$ C^∞-maps | $\mathfrak{C}$ the plane projection of $\mathfrak{C}_n$

such that:

- $\mathfrak{C}_n$ smooth (actually, full-rank Jacobian).
- Bad points set is finite.
- At most two bad points have the same projection.

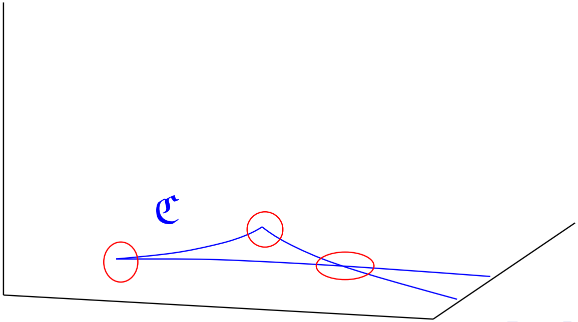


Assumptions

$\mathfrak{C}_n \subset \mathbb{R}^n$: zero set of $n-1$ C^∞-maps | $\mathfrak{C}$ the plane projection of $\mathfrak{C}_n$

such that:

- $\mathfrak{C}_n$ smooth (actually, full-rank Jacobian).
- Bad points set is finite.
- At most two bad points have the same projection.
- Bad points project to nodes or ordinary cusps.



Results

Theorem 1

The previous assumption is generic.

Results

Theorem 2

Under the previous assumption, there exists a square regular system in $\mathbb{R}^{2n-1}$ ($\times 2$ $\# var$) that characterizes the singularities of the $2D$ curve.

Results

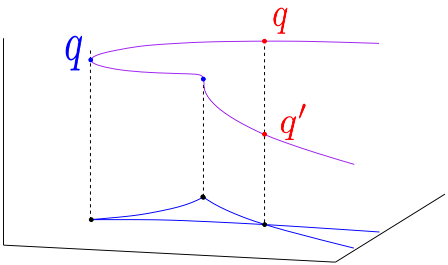
Theorem 2

Under the previous assumption, there exists a square regular system in $\mathbb{R}^{2n-1}$ ($\times 2$ #*var*) that characterizes the singularities of the 2D curve.

For $q \in \mathfrak{C}_n$:

$\pi(q)$ node $\iff$ $\pi(q) = \pi(q')$ → $\times 2$ #*var*

$\pi(q)$ cusp $\iff$ tangent of $\mathfrak{C}_n$ at q is orthogonal → $\times 2$ #*var*



Definition

Let x, r be two sets of $n-2$ real variables and t be a single real variable. For an analytic map $f : U \to \mathbb{R}$, with $U \subseteq \mathbb{R}^n$, we define the maps:,

$$S \cdot f(x_1, x_2, x, r, t) = \left\{ \begin{array}{ll} \frac{1}{2}[\, f(x_1, x_2, x + r\sqrt{t}) + f(x_1, x_2, x - r\sqrt{t})\,], & \text{for } t \neq 0 \\ f(x_1, x_2, x), & \text{for } t = 0 \end{array} \right\}$$

and

$$D \cdot f(x_1, x_2, x, r, t) = \left\{ \begin{array}{ll} \frac{1}{2\sqrt{t}}[\, f(x_1, x_2, x + r\sqrt{t}) - f(x_1, x_2, x - r\sqrt{t})\,], & \text{for } t \neq 0 \\ \nabla f \cdot (0, 0, r), & \text{for } t = 0. \end{array} \right\}$$

If $g(x) \in C^\infty +$ even $\to g(\sqrt{x}) \in C^\infty$

$\mathfrak{C}_n : P_1 = \cdots = P_{n-1} = 0$ satisfies the assumption.

Theorem 2 in more details

$(x_1, x_2) \in \mathbb{R}^2$ is a singular point in $\pi(C_P)$ if and only if there exists a solution of the following system of the form $(x_1, x_2, x, r, t) \in \mathbb{R}^{2n-1}$

$$\left\{ \begin{array}{l} S \cdot P_1(x_1, x_2, x, r, t) = \cdots = S \cdot P_{n-1}(x_1, x_2, x, r, t) = 0 \\ D \cdot P_1(x_1, x_2, x, r, t) = \cdots = D \cdot P_{n-1}(x_1, x_2, x, r, t) = 0 \\ |r|^2 = 1 \end{array} \right\}.$$

Theorem 2 in more details

Moreover, the same system is regular at its solutions.

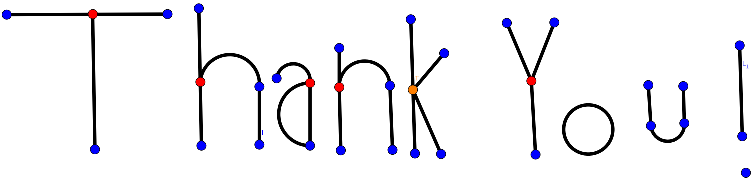
$$\left\{ \begin{array}{l} S \cdot P_1(x_1, x_2, x, r, t) = \cdots = S \cdot P_{n-1}(x_1, x_2, x, r, t) = 0 \\ D \cdot P_1(x_1, x_2, x, r, t) = \cdots = D \cdot P_{n-1}(x_1, x_2, x, r, t) = 0 \\ |r|^2 = 1 \end{array} \right\}.$$

Summary

- Efficient certified methods to isolate singularities of plane curves.
- Not only polynomials but also smooth maps.
- Assumption+No cusp $\rightarrow$ stable singularities: approximation of $\mathfrak{C}_n$ gives the same topology of $2D$ curve.

Future work

- Checking the assumption efficiently.
  - Polynomials: done
  - More general maps: ?
- Proving that a generic silhouette curve satisfies our assumption
- Implementing the algorithms
- Computing the topology of generic singular surfaces in $\mathbb{R}^n$



Beltrán, C. and Leykin, A. (2013). Robust certified numerical homotopy tracking. *Foundations of Computational Mathematics*, 13(2):253–295.

Liang, C., Mourrain, B., and Pavone, J.-P. (2008). Subdivision methods for the topology of 2d and 3d implicit curves. In *Geometric modeling and algebraic geometry*, pages 199–214. Springer, Berlin.

Lin, L. and Yap, C. (2011). Adaptive isotopic approximation of nonsingular curves: The parameterizability and nonlocal isotopy approach. *Discrete Comput. Geom.*, 45(4):760–795.

Plantinga, S. and Vegter, G. (2004). Isotopic approximation of implicit curves and surfaces. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 245–254, New York, NY, USA. ACM.

Snyder, J. M. (1992). Interval analysis for computer graphics. *SIGGRAPH Comput. Graph.*, 26(2):121–130.